View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Jurnal Online Universitas Muhammadiyah Purwokerto

Jurnal Riset Sains dan Teknologi
Volume 3 No. 2 September 2019, 85 - 92
e-ISSN: 2549-9750
p-ISSN: 2579-9118

Pengujian Algoritma *Load Balancing* pada Web Server Menggunakan NGINX

Testing Web Server Load Balancing Algorithm Using NGINX

Dimara Kusuma Hakim[1,*], Dwi Yoga Yulianto[2], Achmad Fauzan[3]
[1,2,3)]Teknik Informatika, Fakultas Teknik dan Sains, Universitas Muhammadiyah Purwokerto
Jl. Raya Dukuhwaluh, Purwokerto 53182
email: [1,*]dimarakusumahakim@gmail.com, [2]dwiyoga.y23@gmail.com, [3]mr.achmadfauzan@gmail.com

DOI:
10.30595/jrst.v3i2.5165

Histori Artikel:

Diajukan:
14/08/2019

Direvisi:
22/09/2019

Diterima:
27/09/2019

ABSTRAK

Peningkatan jumlah *traffic* menyebabkan kerja *web server* yang melayani permintaan menjadi semakin berat. Akibatnya performa *server* menurun dan sering terjadi gangguan pada layanan-layanan *web* tersebut. Sistem Informasi Layanan Desa Kabupaten Purworejo adalah salah satu aplikasi berbasis *website* yang berfungsi untuk menyimpan dan mengelola informasi serta layanan desa yang ada di kabupaten Purworejo. Kurang stabilnya kecepatan akses *website* ini menjadi salah satu kendala yang dihadapi oleh pengguna saat bekerja dengan *website* ini. Dengan pemanfaatan sistem operasi *Ubuntu* dan penerapan virtualisasi di Oracle VM VirtualBox serta menggunakan *load balancing web server* Nginx, beban *web server* saat terjadi *request* yang tinggi karena banyaknya pengguna yang mengakses sekaligus dalam kurun waktu yang sama dapat dibagi agar beban *traffic web server* tidak terlalu berat. Dalam penelitian ini hasil penggunaan algoritma *least connection* untuk *load balancing* lebih memberikan performa yang baik dibandinganken algoritma *round robin* terhadap kecepatan akses *website* saat terjadi *request* yang sangat banyak dari pengguna dalam waktu yang bersamaan.

Kata kunci: *Web server, Concurrent Access, C10K Problem , Nginx, Load balancing*

ABSTRACT

Increasing the amount of traffic causes the work of the server serving the request becomes increasingly heavy. As a result, server performance decreases and frequent interruptions in these web services. The village service information system in Purworejo district is a website based application that functions to store and manage village information and services in Purworejo district. The lack of stable speed of access to this website is one of the obstacles faced by users when working with this website. With the use of the Ubuntu operating system and the application of virtualization in Oracle VM VirtualBox and using the Nginx web server load balancing, web server loads when requests are high because many users access at once in the same time period can be shared so that web server traffic loads are not too heavy. In this study the results of using the least connection algorithm for load balancing provide better performance compared to the round robin algorithm on the website access speed when there are very many requests from users at the same time.

Keywords: Web server, Concurrent Access, C10K Problem , Nginx, Load balancing.

1. PENDAHULUAN

Seiring berkembangnya teknologi, jumlah penggunaan layanan web semakin meningkat. Hal ini menyebabkan situs-situs web memiliki jumlah *traffic* yang tinggi. Kegiatan atau acara tertentu juga dapat menyebabkan naiknya jumlah *traffic* web suatu organisasi. Peningkatan jumlah *traffic*

menyebabkan kerja server yang melayani permintaan menjadi semakin berat. Akibatnya performa server menurun dan sering terjadi gangguan pada layanan-layanan web tersebut. Jika tidak ditangani, hal ini dapat mengakibatkan sistem ataupun situs web tersebut mati/ down.

Melihat tingkat aktivitas dari pengguna yang begitu tinggi, tentu saja hal ini akan berdampak pada penyedia informasi. Kinerja server web dan database sebagai media penyedia konten selalu diharapkan dapat memenuhi semua kebutuhan dari pengguna. Jika tidak ditanggapi dengan serius, ini bisa saja berakibat pada server-server yang kelebihan beban permintaan (*request*) dari pengguna. Hal ini disebabkan permintaan dari pengguna lebih besar dari kemampuan server untuk memberikan layanan. Dampak ini tentu tidak diinginkan oleh beberapa instansi yang semua aktivitasnya sudah ketergantungan dengan jaringan komputer. Seperti misal masalah C10K yaitu masalah mengoptimalkan soket jaringan untuk menangani sejumlah besar klien secara bersamaan. Dengan kata lain, menangani permintaan per detik berkaitan dengan kecepatan menangani permintaan, sedangkan sistem yang mampu menangani sejumlah besar koneksi *concurrent access* tidak harus menjadi sistem yang cepat, hanya satu di mana setiap permintaan secara deterministik akan mengembalikan respon. dalam jumlah waktu (belum tentu tetap) terbatas.

Salah satu solusi untuk menangani masalah tersebut adalah dengan menggunakan sebuah server yang andal dengan performa yang tinggi. Solusi lain yang dapat diterapkan adalah teknologi *load balancing*. Teknologi *load balancing* dapat membagi beban permintaan ke beberapa web server sehinga meringankan kinerja masing-masing server (Dani & Suryawan, 2017).

Menurut Pandey, *et al*. (2015) *load balancing* merupakan aspek penting yang dapat mendistribusikan beban kerja ke banyak server secara optimal yang menghasilkan waktu tanggap yang baik dan meningkatkan tingkat kepuasaan pengguna, meningkatkan efisiensi penggunaan sumber daya, dan berdampak pada peningkatan performa secara keseluruhan.

Menurut Aziz & Tampati (2015) *load balancing* menggunakan Nginx menjadi pilihan berkat kinerjanya yang tinggi, stabilitas, kekayaan akan fitur, mudah dikonfigurasi dan terutama hemat sumber daya.

Pada penelitian yang di lakukan oleh Singh & Kumar (2015) mengimplementasikan load balancing dari metode algoritma web server queueing dengan cara kerja web server yang kelebihan beban telah dikelola menggunakan berbagai teknik load balancing yang terbuka untuk tantangan bagi para peneliti. Ada beberapa algoritma yang ada untuk mengelola beban web server berdasarkan pada transfer beban dari satu node web server ke yang lain. Pendekatan transfer beban web server meningkatkan pemanfaatan server dan meminimalkan waktu respon. Beberapa algoritma load balancing statis dan dinamis telah dipertimbangkan untuk berbagai estimasi beban server berdasarkan faktor-faktor seperti beban memori, jumlah koneksi, dan perilaku web server.

Menurut Rahmatulloh & Nursuwars (2017) *load balancing* adalah sebuah teknik untuk mendistribusikan beban trafik pada dua atau lebih jalur koneksi secara seimbang, agar trafik dapat berjalan optimal, memaksimalkan *throughput*, memperkecil waktu tanggap dan menghindari *overload* pada salah satu jalur koneksi. *Load balancing* digunakan pada saat sebuah *server* telah memiliki jumlah user yang telah melebihi maksimal kapasitasnya. *Load balancing* juga mendistribusikan beban kerja secara merata di dua atau lebih komputer, link jaringan, CPU, hard drive, atau sumber daya lainnya, untuk mendapatkan pemanfaatan sumber daya yang optimal.

Sedangkan menurut Membrey, et al. (2012) menjelaskan fungsi dari load balancing dari kasus perusahaan besar yang menjual berbagai jenis mainan berbulu secara online. Masalahnya adalah bahwa seperti kebanyakan bisnis mereka memiliki periode sibuk dan sunyi. Rata-rata mereka akan mengambil sekitar 2.000 pesanan per minggu. Namun, pada waktu-waktu tertentu. Server mereka tidak bisa menangani beban sebanyak ini. Dengan banyak pesanan itu, kita bisa membayangkan berapa banyak halaman web dan grafik yang perlu dihasilkan dan dikirim kembali ke pelanggan. Untuk setiap pesanan yang dilakukan, puluhan orang bisa menjelajahi situs web. Selama periode normal, server mereka benar-benar tidak punya masalah mengatasi beban. Namun, selama periode tinggi, server akan mulai mengalami masalah.

Penelitian sebelumnya yang dilakukan oleh Dani dan Suryawan (2017) mereka melakukan perancangan dan pengujian *load balancing* dan failover menggunakan nginx

untuk *load balancing* dan KeepAlived untuk failover menghasilkan beberapa sistem yang dirancang dapat membagikan beban secara merata ke beberapa backend server baik dalam keadaan semua server normal atau pun saat terjadi kegagalan pada salah satu backend server.

Selanjutnya Devi & Uthariaraj (2016) melakukan penelitian load balancing di lingkungan cloud computing menggunakan algoritma round robin untuk peningkatan nonpreemptive dependent tasks. Dalam kinerja ini, algoritma round robin yang ditingkatkan ini paling cocok untuk yang heterogen/homogen pekerjaan dengan sumber daya heterogen (VM) dibandingkan dengan algoritma round robin dan round robin lainnya. Algoritma ini menganggap waktu respons sebagai QoS utama parameter.

Berdasarkan analisa permasalahan diatas maka perlu untuk mengimplementasikan *load balancing* pada web server menggunakan Nginx beserta algoritma yang diuji yaitu algoritma *round robin* dan algoritma *least connection* untuk menentukan algoritma yang baik dalam kinerja sistem *load balancing* pada web server aplikasi sistem informasi layanan data desa kabupaten Purworejo.

Tujuan dari penelitian ini adalah untuk menguji kinerja *load balancer* tiap server pada Nginx sebagai server *load balancing* guna membagi beban *traffic* web server menggunakan Ubuntu pada aplikasi sistem informasi layanan desa kabupaten Purworejo sehingga dapat mengoptimalkan penggunaan web server untuk menghasilkan kecepatan akses yang stabil dan merata pada aplikasi *website* saat dijalankan oleh banyak pengguna dalam waktu yang bersamaan.

Seperti yang sudah dijelaskan bahwa *load balancing* adalah proses pendistribusian beban secara merata pada beberapa komputer. Sistem *load balance* bertujuan untuk menyeimbangkan pembagian kerja di antara node-node yang ada di dalam sebuah cluster.

Hal tersebut tidak jauh berbeda yang dijelaskan oleh Bourke (2001), *load balancing* merupakan sebuah proses dan teknologi yang menyalurkan lalu lintas (*traffic*) diantara beberapa server menggunakan perangkat berbasis jaringan. Perangkat ini (*load balancing* server) menahan/menangkap *traffic* yang bertujuan kepada sebuah alamat kemudian redirect *traffic* tersebut kepada banyak server. Proses *load balance* ini bersifat transparan terhadap pengguna yang melakukan *request* ke server *load balance*. Ada puluhan bahkan ratusan server yang beroperasi dibalik sebuah alamat.

Nginx adalah software open-source yang memiliki kinerja tinggi sebagai server HTTP dan reverse proxy. Nginx dengan cepat memberikan konten statis dengan penggunaan efisien sumber daya sistem. Hal ini dapat menyebarkan dinamis HTTP konten di jaringan

menggunakan FastCGI handler untuk script, dan dapat berfungsi sebagai perangkat lunak yang sangat mampu penyeimbang beban. Nginx dapat dijalankan dan tersedia untuk platform Unix, Linux, varian dari BSD, MacOS X, Solaris, dan Microsoft Windows) (Aziz dan Tampati, 2015).

Sedangkan Afrianto, et al. (2018) melakukan penimbangan dengan menggunkan metode load balancing round robin untuk meningkatkan web server cluster di jaringan OpenFlow menghasilkan bahwa WRR di jaringan OpenFlow mengalami penurunan throughput sebesar 2% dan telah meningkatkan koneksi yang hilang 49%. Namun, itu juga dapat meningkatkan waktu respons hingga 57% dan koneksi HTTP berhasil 10%.

Sedangkan menurut Irza, et al. (2017) menjelaskan Nginx didirikan atas dasar event-base architecture (EBA). Kompnen berinteraksi secara khusus dengan menggunakan event notification, bukan metode panggilan langsung. Event Notification ini terjadi dari tugas yang berbeda. Kemudian diantrikan untuk diproses oleh event handler (pengendali event). Event Handler berjalan secara berulang, ketika event tersebut diproses, kemudian dikeluarkan dari antrian dan kemudian dilanjutkan ke proses selanjutnya. Dengan demikian, pekerjaan yang dilakukan oleh sebuah thread (jalur) sangat mirip dengan scheduler (agenda), persilangan beberapa koneksi ke satu aliran eksekusi.

Algoritma *round robin* yaitu algoritma ini menggilir proses yang ada di antrian. Proses akan mendapat jatah sebesar *time quantum*. Jika *time quantum*-nya habis atau proses sudah selesai, CPU akan dialokasikan ke proses berikutnya. Tentu proses ini cukup adil karena tak ada proses yang diprioritaskan, semua proses mendapat jatah waktu yang sama dari CPU yaitu (1/n), dan tak akan menunggu lebih lama dari (n-1)q dengan q adalah lama 1 *quantum*. Algoritma ini sepenuhnya bergantung besarnya *time quantum*. Jika terlalu besar, algoritma ini akan sama saja dengan algoritma *first come first served*. Jika terlalu kecil, akan semakin banyak

peralihan proses sehingga banyak waktu terbuang. (Wijaya & Gunawan, 2018).

Sedangkan Algoritma *Least connection* melakukan pembagian beban berdasarkan banyaknya koneksi yang sedang dilayani oleh sebuah server. Server dengan koneksi yang paling sedikit akan diberikan beban berikutnya, begitu pula server dengan koneksi banyak akan dialihkan bebannya ke server lain yang bebannya lebih rendah (Ellrod, 2010).

2. METODE PENELITIAN

Dalam penelitian ini jenis penelitian yang digunakan adalah jenis penelitian metode campuran (*mixed method*) yaitu penelitian pengembangan dengan pengujian kuantitatif dan kualitatif (menggunakan uji statistik).

Uji statistik digunakan untuk mengetahui perbedaan rata-rata dua populasi/kelompok data yang independen dalam hal ini pada algoritma *round robin* dan algoritma *least connection*, serta membandingkan kecepatan akses dari dua algoritma tersebut.

2.1. Variabel Yang Diteliti

Variabel yang diteliti dalam object penelitian aplikasi sistem informasi layanan desa kabupaten Purworejo adalah sebagai berikut:

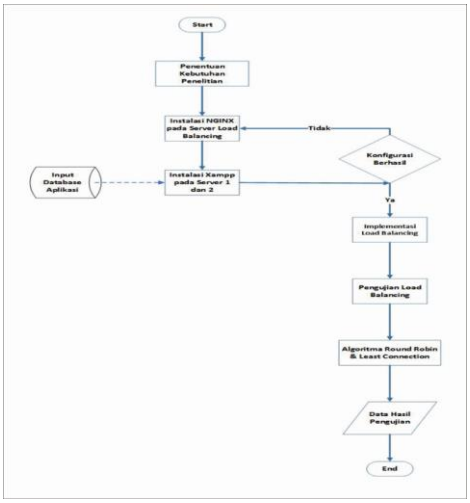
1. Halaman Tampil Layanan Desa Kabupaten Purworejo Menguji kecepatan akses *website* saat terjadi *request select* data layanan tiap desa pada aplikasi *website.*
2. Halaman Data Laporan Desa Kabupaten Purworejo Menguji kecepatan akses *website* saat terjadi *request* menampilkan halaman data laporan desa kabupaten Purworejo.

2.2. Sumber Data

Dalam penelitian ini, metode pengumpulan data yang diambil adalah dengan melakukan pengamatan pada *website* aplikasi sistem informasi layanan data desa kabupaten Purworejo untuk menganalisa kebutuhan, permasalahan yang muncul, kecepatan akses dan topologi jaringan yang sudah ada saat ini. Kemudian melakukan konsep *load balancing* web server dengan menggunakan algoritma *round robin* dan algoritma *least connection*.

2.3. Alur Pengembangan Jaringan

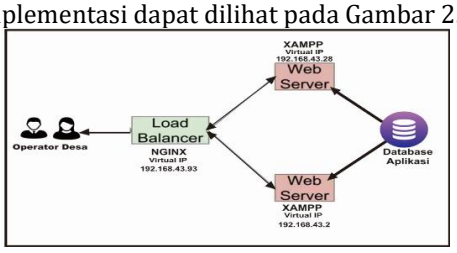
Alur pengembangan sistem jaringan yang dijelaskan pada Gambar 1.



Gambar 1. Alur Pengembangan Jaringan

2.4. Desain Topologi Jaringan

Tahap ini dilakukan percobaan terhadap topologi. Topologi dibuat dengan pemberian IP Address berbasis DHCP masing-masing mesin virtual. Pecobaan yang dilakukan sebelum memasuki tahap implementasi dapat dilihat pada Gambar 2.



Gambar 2. Desain Topologi Jaringan

Dalam penelitian ini dilakukan pengembangan jaringan dengan Network Development Life Cycle (NDLC). Menurut Kosasi (2011), NDLC merupakan suatu siklus hidup pengembangan sistem jaringan komputer yang bersifat komprehensif dengan tingkat integritas yang kuat dari sejumlah tahapan yang harus dilakukan untuk mencapai sebuah keluaran yang akurat, valid dan memiliki produktivitas yang tinggi.

2.5. Analisis Data

Penelitian ini menguji *load balancing* web *server* pada aplikasi sistem informasi layanan desa kabupaten Purworejo. Pengujian ini dilakukan 4 kali agar mengetahui hasil algoritma yang lebih baik diterapkan untuk kecepatan akses *website* dengan tahapan antara lainnya:

1. Pengujian rata-rata kecepatan akses menggunakan algoritma *Round Roubin* saat ada perintah menampilkan halaman tampil layanan dan data laporan desa kabupaten Purworejo. Serta pengujian rata-rata

kecepatan akses menggunakan algoritma *Least connection* saat ada perintah menampilkan halaman tampil layanan dan data laporan desa kabupaten Purworejo. Rumus rata-rata (*mean*) yang digunakan adalah sebagai berikut:

$$\bar{x} = \frac{x_1 + x_2 + x_3 + \cdots + x_n}{n}$$

Keterangan : $\bar{x}$ = Rata-rata
x = Nilai ke-i
n = Jumlah data

2. Pengujian kecepatan akses *website* menggunakan *software* Webserver Stress Tool.

3. Melakukan uji hipotesis dengan menggunakan software SPSS dari algoritma *round robin* dan algoritma *least connection* pada layanan dan laporan dsa aplikasi sistem informasi layanan data desa.

4. Menganalisis algoritma *load balancing* web server yaitu algoritma *round robin* dan algoritma *least connection* yang lebih tepat digunakan untuk mengakses kecepatan aplikasi sistem informasi layanan desa kabupaten Purworejo.

3. Hasil dan Pembahasan

3.1. Implementasi *Load Balancing* Web Server Nginx

Nginx diinstall pada *server load balancing* pada OS Ubuntu. Nginx digunakan sebagai *software load balancing* untuk membagi beban komputasi pada web server. Berikut ini adalah langkah-langkah setting Nginx.

a. *Install* Nginx dengan perintah: "sudo apt-get isntall –y nginx" (tanpa tanda petik), seperti pada Gambar 3.

```
root@load-balance:/home/load-balance# apt-get install -y nginx
```

Gambar 3. *Install* Nginx

b. Untuk mengecek status Nginx *running* dengan cara perintah: "systemctl status nginx" (tanpa tanda kutip), seperti pada Gambar 4.

```
root@load-balance:/home/load-balance# systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: en
   Active: active (running) since Wed 2019-07-17 02:21:57 WIB; 8min ago
     Docs: man:nginx(8)
  Process: 690 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=
  Process: 678 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process o
 Main PID: 691 (nginx)
    Tasks: 2 (limit: 1711)
   CGroup: /system.slice/nginx.service
           ├─691 nginx: master process /usr/sbin/nginx -g daemon on; master_proc
           └─692 nginx: worker process

Jul 17 02:21:57 load-balance systemd[1]: Starting A high performance web server
Jul 17 02:21:57 load-balance systemd[1]: nginx.service: Failed to parse PID from
Jul 17 02:21:57 load-balance systemd[1]: Started A high performance web server a
lines 1-15/15 (END)
```

Gambar 4. Mengecek Status Nginx

c. Selanjutnya masuk ke halaman konfigurasi nginx dengan cara ketik perintah " nano /etc/nginx/sites-available/default" (tanpa tanda kutip). Seperti Gambar 5.

```
root@load-balance:/home/load-balance# nano /etc/nginx/sites-available/default
```

Gambar 5. Perintah Membuka Konfigurasi Nginx

d. Konfigurasi Nginx dengan menambahkan perintah pada halaman konfigurasi paling bawah, seperti pada Gambar 6 yaitu dengan algoritma *round roubin* dan pada Gambar 7 yaitu algoritma *Least Connection*.

```
upstream backend {
        server 192.168.43.28; #server 1
        server 192.168.43.2;  #server 2
}

server {
        listen 80;
        location /{
                proxy_pass http://backend;
        }
}
```

Gambar 6. Konfigurasi Nginx Algoritma *Round Roubin*

```
upstream backend {
        least_conn;
        server 192.168.43.28; #server 1
        server 192.168.43.2;  #server 2
}

server {
        listen 80;
        location /{
                proxy_pass http://backend;
        }
}
```

Gambar 7. Konfigurasi Nginx Algoritma *Least Connection*

3.2. Pengujian Algoritma *Round Roubin*

Pengujian ini dilakukan dengan semua pengguna disimulasikan mengakses *website* aplikasi sistem informasi layanan desa kabupaten Purworejo menggunakan 1 server *load balancing* dengan algoritma *Round Roubin*, yaitu 494 pengguna mengakses secara bersamaan dan melihat hasil kecepatan waktu eksekusi halaman dari tiap-tiap pengguna. *Request* halaman yang diuji adalah menampilkan halaman layanan data desa dan halaman laporan data desa. Hasil pengujian rata-rata kecepatan akses menggunakan Webserver Stress Tool dijelaskan pada Tabel 1 dan Tabel 2.

Tabel 1. Hasil Pengujian Rata-Rata Kecepatan Akses Menggunakan Algoritma *Round robin* pada Layanan Data Desa

Clicks/User	Errors [%]	*Time* Spent [ms]	Avg. Click *Time* [ms]
494	0,00	1.310.745	2.653

Tabel 2. Hasil Pengujian Rata-Rata Kecepatan Akses Menggunakan Algoritma *Round robin* pada Laporan Data Desa

Clicks/User	Errors [%]	*Time* Spent [ms]	Avg. Click *Time* [ms]
494	0,00	1.080.418	2.187

3.3. Pengujian Algoritma *Least connection*

Pengujian ini dilakukan dengan semua pengguna disimulasikan mengakses *website* aplikasi sistem informasi layanan desa kabupaten Purworejo menggunakan 1 server *load balancing* dengan algoritma *Least connection*, yaitu 494 pengguna mengakses secara bersamaan dan melihat hasil kecepatan waktu eksekusi halaman dari tiap-tiap pengguna. *Request* halaman yang diuji adalah menampilkan halaman layanan data desa dan halaman laporan data desa. Hasil pengujian menggunakan Webserver Stress Tool dapat dilihat pada Tabel 3 dan Tabel 4.

Tabel 3. Hasil Pengujian Rata-Rata Kecepatan Akses Menggunakan Algoritma *Least connection* pada Layanan Data Desa

Clicks/User	Errors [%]	*Time* Spent [ms]	Avg. Click *Time* [ms]
494	0,00	1.131.557	2.291

Tabel 4. Hasil Pengujian Rata-Rata Kecepatan Akses Menggunakan Algoritma *Least Connection* Pada Laporan Data Desa

Clicks/User	Errors [%]	*Time* Spent [ms]	Avg. Click *Time* [ms]
494	0,00	394.684	799

3.4. Uji Rata-Rata 2 Populasi Independen

Untuk mengukur tingkat efektifitas dari proses virtualisasi dengan *load balancing* ini, maka dilakukan uji t atau uji rata-rata. Uji ini bertujuan untuk membandingkan rata-rata dari algoritma *round roubin* dan *least connection* yang tidak berhubungan satu sama lain, apakah kecepatan akses saat menggunakan algoritma *round robin* dan *least connection* tersebut mempunyai rata-rata yang sama, ataukah tidak secara signifikan.

Uji rata-rata (Independent Samples T Test), uji ini bertujuan untuk membandingkan rata-rata dari dua grup yang tidak berhubungan satu dengan yang lain, apakah kedua grup tersebut mempunyai rata-rata yang sama, ataukah tidak secara signifikan (Mustafidah & Taniredja 2011).

Melakukan uji rata-rata 2 populasi independen menggunakan *software* SPSS. Tujuan dari pengujian ini yaitu memandingkan rata-rata kecepatan akses *website*, apakah ada perbandingan yang signifikan antara penggunaan *round roubin* dan *least connection* ataukah tidak secara signifikan.

a. Perumusan Hipotesis
   H_0 : $\mu_1 = \mu_2$; Tidak ada perbedaan kecepatan akses yang signifikan antara algoritma *round robin* dan *least connection*.
   H_1: $\mu_1 \neq \mu_2$; Ada perbedaan kecepatan akses yang signifikan antara algoritma *round robin* dan *least connection*.
b. Penentuan α
   α = 5% (0,05)
   Nilai α = 5% (0,05) adalah tingkat signifikan yang menjadi patokan dalam hal ini berarti mengambil resiko salah dalam mengambil keputusan untuk menolak hipotesis yang benar sebanyak-banyaknya 5% (0,05) adalah ukuran standar yang sering digunakan dalam penelitian.
c. Penentuan Simpulan
   Penerimaan H_0 yang berarti tidak ada perbedaan kecepatan akses yang signifikan antara algoritma *round robin*

dan *least connections* atau penolakan H_0 yang berarti ada perbedaan kecepatan akses yang signifikan antara algoritma *round robin* dan *least connections.*

Berikut ini adalah adalah uji rata-rata 2 populasi independen, seperti pada Gambar 8 dan Gambar 9.

a. *Request* Tampil Layanan Data Desa

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
									Lower	Upper
Hasil Algoritma Nginx	Equal variances assumed	211,499	0,000	5,126	986	0,000	362,73077	70,76834	223,85493	501,60661
	Equal variances not assumed			5,126	881,470	0,000	362,73077	70,76834	223,83025	501,63128

Gambar 8. Hasil Pengujian Rata-Rata 2 Populasi Independen Pada Tampil Layanan Data Desa

Berdasarkan Gambar 8 diketahui bahwa nilai signifikasi = 0,000 yang berarti < α (0,05 atau 5%), maka H_0 ditolak, artinya ada perbedaan signifikan antara algoritma *round robin* dengan *least connection* dalam hal kecepatan akses saat menampilkan layanan data desa. Selisih rata-rata kecepatan aksesnya adalah 2653,3300 – 2290,5992 = 362,7308, dengan menggunakan algoritma *least connection* akan lebih cepat dengan selisih 362,7308 ms dibandingkan algoritma *round robin*. Jika dilihat dari Interval Konvidensinya (*Confidence Interval of the Difference*) atau rentan antara hasil rata-rata dua algoritma berada pada daerah *positive* baik bagian *Lower* maupun *Upper*-nya, sehingga bisa dikatakan rata-rata kecepatan akses saat *request* tampil layanan data desa dengan algoritma *round robin* > rata-rata kecepatan akses saat *request* tampil layanan data desa dengan algoritma *least connection*. Artinya menggunakan algoritma *least connection* memberikan kecepatan akses yang lebih baik daripada menggunakan algoritma *round robin*.

b. *Request* Tampil Laporan Data Desa

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
									Lower	Upper
Hasil Algoritma Nginx	Equal variances assumed	371,205	0,000	29,672	986	0,000	1338,10231	45,78150	1248,36656	1427,83806
	Equal variances not assumed			29,672	646,665	0,000	1338,10231	45,78150	1248,24732	1427,97730

Gambar 9. Hasil Pengujian Rata-Rata 2 Populasi Independen Pada Laporan Data Desa

Berdasarkan Gambar 9 diketahui bahwa nilai signifikasi = 0,000 yang berarti < α (0,05 atau 5%), maka Ho ditolak, artinya ada perbedaan signifikan antara algoritma *round robin* dengan *least connection* dalam hal kecepatan akses saat terjadi *request* tampil laporan data desa bulanan. Selisih rata-rata kecepatan aksesnya adalah 2187,0769 – 798,9676 = 1388,1093, dengan menggunakan algoritma *least connection* akan lebih cepat dengan selisih 1388,1093 ms dibandingkan algoritma *round robin*. Jika dilihat dari Interval Konvidensinya (*Confidence Interval of the Difference*) atau rentan antara hasil rata-rata dua algoritma berada pada daerah *positive* baik bagian *Lower* maupun *Upper*-nya, sehingga bisa dikatakan rata-rata kecepatan akses saat *request* tampil laporan data desa dengan algoritma *round robin* > rata-rata kecepatan akses saat *request* tampil laporan data desa dengan algoritma *least connection*. Artinya menggunkan algoritma *least connection* memberikan kecepatan akses yang lebih baik daripada menggunakan algoritma *round robin*.

4. KESIMPULAN

Dari hasil analisis pengujian menggunakan uji rata-rata kecepatan akses dari 2 populasi independen, yaitu dari algoritma round roubin dan *least connection* didapatkan hasil bahwa penggunaan algoritma *least connection* untuk *load balancing* lebih memberikan performa yang baik dibandinganken algoritma *round robin* terhadap kecepatan akses *website* saat terjadi *request* yang sangat banyak dari pengguna dalam waktu yang bersamaan. Hal ini tentunya sangat tepat untuk membagi beban *traffic* web server yang bertujuan agar akses *website* tetap berjalan dengan cepat dan stabil sehingga memudahkan pengguna dalam melakukan pekerjaannya.

DAFTAR PUSTAKA

Afrianto, Y., Sukoco, H., & Wahjuni, S. (2018). Weighted Round Robin Load Balancer to Enhance Web Server Cluster in OpenFlow Networks. TELKOMNIKA (Telecommunication Computing Electronics and Control), 16(3), 1402–1408.

Aziz, A., & Tampati, T. (2015). Analisis Web Server untuk Pengembangan Hosting Server Institusi: Pembandingan Kinerja Web Server Apache dengan Nginx. *Multinetics*, *1*(2), 12–20.

https://doi.org/10.32722/multinetics.vol1.no.2.2015.pp.12-20.

Bourke, T. (2001). *Server Load Balancing* (J. Sumser, ed.). O'Reilly.

Dani, R., & Suryawan, F. (2017). Perancangan dan Pengujian Load BALANCING DAN FAILOVER MENGGUNAKAN Nginx. *Khazanah Informatika*, *3*(1), 43–50.

Devi, D. C., & Uthariaraj, V. R. (2016). Load Balancing in Cloud Computing Environment Using Improved Weighted Round Robin Algorithm for Nonpreemptive Dependent Tasks. Scientific World Journal, 1–14.

Ellrod, C. (2010). Load Balancing–Round Robin. Retrieved 2015, from http://blogs.citrix.com/2010/09/03/load-balancing-round-robin. (Accessed: 14 June 2019).

Irza, I. F., Zulhendra, & Efrizon. (2017). Analisis Perbandingan Kinerja Web Server Apache dan Nginx Menggunakan Httperf Pada Portal Berita (Studi Kasus beritalinux.com). Jurnal Vokasional Teknik Elektronika & Informatika, 5(2), 75–82.

Kosasi, S. (2011). Penerapan Network Development Life Cycle Untuk Pengembangan Teknologi Thin Client. Jurnal Ilmiah Komputasi Dan Elektronika, 1(2), 125–141.

Membrey, P., Plugge, E., & Hows, D. (2012). Practical Load Balancing Ride the Performance Tiger. New York: Apress.

Mustafidah, H., & Taniredja, T. (2011). Penelitian Kuantitatif (Sebuah Pengantar). Purwokerto, Alfabeta CV.

Pandey, S., Prasanna, S., Kapil, S., & S, R. B. (2015). Load Balancing Techniques : A Comprehensive Study. International Journal of Advance Research in Computer Science and Management Studies, 3(4), 331–335.

Rahmatulloh, A., & Nursuwars, F. M. S. (2017). Implementasi Load Balancing Web Server menggunakan Haproxy dan Sinkronisasi File pada Sistem Informasi Akademik Universitas Siliwangi. Jurnal Teknologi Dan Sistem Informasi, 3(2), 241–248.

Singh, H., & Kumar, S. (2015). WSQ: Web Server Queueing Algorithm for Dynamic Load Balancing. Wireless Personal Communications, 80(1), 229–245.

Wijaya, A., & Gunawan. (2018). Implementasi Algoritma Round Robin Pada Sistem Penjadwalan Mata Kuliah (Studi Kasus : Universitas Muhammadiyah Bengkulu). *Jurnal Informatika Upgris (JIU)*, *4*(1), 64–71.